

1.

By making direct reference to the technologies used, explain how a virtual private network (VPN) allows a travelling salesperson to connect securely to their company's network. [4]

*Award [1] for identifying a technology and [1] for explaining it, for **two** technologies, up to [4 max].*

Tunnelling protocols;

Allows the data to be encapsulated/hidden whilst travelling across the internet;

Encryption protocols (IPSEC);

If hacked it will not be understandable;

The use of gateways;

Allows the salesperson to connect with the company's server; [4]

2.

Construct a truth table for the following Boolean expression.

$(A \text{ AND } B) \text{ NOR } C$  [3]

*Award [3] for completely correct table.*

*Award [2] if only 6 or 7 rows are correct.*

*Award [1] if only 4 or 5 rows are correct.*

*Award [0] otherwise, or if table does not contain 8 rows.*

A	B	C	$(A \text{ AND } B) \text{ NOR } C$
0	0	0	1
0	0	1	0
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	0

[3]

3.

A sub-program `all_even()` accepts a positive integer `N` and outputs `true` if all digits of `N` are even, otherwise it outputs `false`. For example, `all_even(246)` outputs `true` and `all_even(256)` outputs `false`.

The following algorithm is constructed for the sub-program `all_even(N)`.

```

EVEN = true
loop while (N > 0) and (EVEN = true)
    if (N mod 10) mod 2 = 1 then
        EVEN = false
    end if
end loop
output EVEN

```

(a) Explain why this algorithm does not obtain the correct result. [2]

(b) Outline what should be changed in the algorithm to obtain the correct result. [3]

(a) *Award up to [2 max].*  
 The value of `N` is never changed;  
 So the logical expression in the while loop always evaluates to true;  
 And loop repeats an infinite number of times; [2]

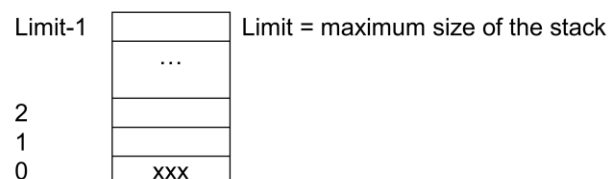
(b) Statement `N = N div 10`;  
 Should be written within the while loop;  
 After the if statement; [3]

4.

(a) Draw an annotated diagram showing how an array can be used to store a stack. [2]

(b) Explain how elements in the stack may be reversed using a queue. [4]

(a) *Award up to [2] max.*  
*Award [1] for the array elements shown,*  
*[1] for either two additional variables or for labelling.*  
*Example answer:*



Variable `Top` is used to hold the subscript of the current top of the stack; [2]

(b) *Award up to [4 max].*  
 Create an empty queue;  
 While stack is not empty;  
     Pop an element from the stack;  
     Enqueue the popped element;  
 While queue is not empty;  
     Dequeue;  
     Push dequeued element back on the stack;

**Note:** Award a maximum of [2] for answers that just describe how a stack and a queue work. [4]

5.

A population study divides a metropolitan area into seven regions: A–G.

The following table shows the current population (in millions) of the regions.

Region	Current population (millions)
A	2.3
B	2.1
C	1.2
D	1.4
E	1.5
F	1.1
G	0.8

Two one-dimensional arrays, `Region` and `Curr_Pop`, are used to hold this data. For example, `Region[0] = 'A'`. The population in region A is 2.3 million and 2.3 is found in `Curr_Pop[0]`.

(a) Construct the algorithm that will output the total population in the metropolitan area. [3]

(a) *Award marks as follows.*

*Award [1] for a correct loop (accept a correct while loop).*

*Award [1] for correct calculation.*

*Award [1] for correct output.*

```
totalpop = 0
loop K from 0 to 6
    totalpop = totalpop + Curr_Pop[K]
end loop
output totalpop
```

[3]

*Note: Award [1] for `Curr_Pop[0] + Curr_Pop[1] + ... + Curr_Pop[6]`  
Award marks only for an algorithm, do not award marks for calculation /  
summation "2.3 + 2.1 + 1.2..."*

The numbers in the following table represent expected **percentages** of yearly migration from one region to another, obtained by analysing historical migration data. For example, it is expected that 0.32 % of the current population of region B will move to region C.

The diagonal entries represent a region's internal growth rate. For example, the population of region C is expected to increase by 1.2 % as a result of the births and deaths of people currently living in region C.

<b>To From</b>	A	B	C	D	E	F	G
A	<b>1.10</b>	0.21	0.21	0.05	0.20	0.20	0.29
B	0.30	<b>1.20</b>	0.32	0.25	0.20	0.09	0.31
C	0.25	0.22	<b>1.20</b>	0.35	0.30	0.23	0.12
D	0.10	0.33	0.36	<b>1.30</b>	0.09	0.12	0.20
E	0.20	0.22	0.24	0.35	<b>1.00</b>	0.20	0.21
F	0.12	0.21	0.13	0.21	0.22	<b>1.40</b>	0.31
G	0.05	0.03	0.30	0.20	0.23	0.26	<b>0.90</b>

- (b) (i) State the **percentage** of the population of region G that are expected to move to region A. [1]
- (ii) Determine the **number** of people from region B who are expected to move to region E. [1]
- (iii) Describe how the change in population of region F in one year could be determined. [3]
- (c) Construct the algorithm that will predict the population in each region after 10 years. You should assume that the yearly migration percentages, given in the table on page 8, remain the same over the 10 years. [7]

- (b) (i) 0.05 (%); [1]
- (ii)  $0.0042 \text{ (million)} / 4200$ ; [1]  
Award [1] for  $0.2 * 2.1 / 100$ .
- (iii) Award up to [3 max].  
Current population in the region F should be increased by the internal growth rate;  
Increased by the population who moved into the region;  
And decreased by the population who moved from this to other regions; [3]  
Accept formulas that correctly address these points.

- (c) Award marks as follows up to **[7 max]**.  
Award **[1]** for correct outer loop (10 years).  
Award **[1]** for correctly initializing Future\_Pop array to zero each year.  
Award **[1]** for correct row loop (k).  
Award **[1]** for correct column loop (j).  
Award **[1]** for correctly calculated internal growth.  
Award **[1]** for correct calculation of population who moved to region, increment.  
Award **[1]** for correct calculation of population who moved from region, decrement.  
Award **[1]** for assignment (new data in Curr\_Pop array is data from Future\_Pop array).

*Example answer:*

```
loop YEAR from 1 to 10
  loop Z from 0 to 6
    Future_Pop[Z]=0
  end loop
  loop K from 0 to 6
    loop J from K to 6
      if J=K then
        Future_Pop[K] = Curr_Pop[K] * Table[K][K]
      else
        Future_Pop[K]=Future_Pop[K]
                      + (Curr_Pop[J]*Table[J][K])
                      - (Table[K][J]*Curr_Pop[K])
      end if
    loop Z from 0 to 6
      Curr_Pop[Z] = Future_Pop[Z]
    end loop
  end loop
end loop
loop Z from 0 to 6
  output Curr_Pop[Z]
end loop
//Output the contents of Curr_Pop array which holds the population
//after 10 years
```

**[7]**

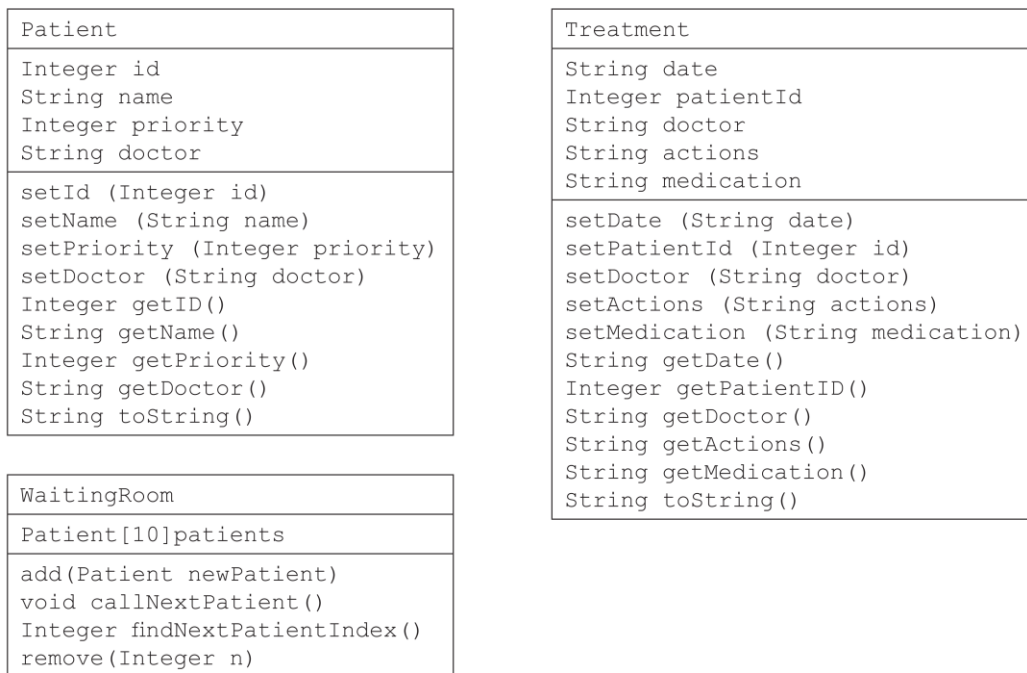
### Option D — Object-oriented programming

A small health clinic with three doctors operates in a village. All clients of the clinic have their details stored in the clinic's database. Patients that visit the clinic during the day are given a priority rating (1–3) and are seated in a waiting room to wait for the next available doctor. When it is their turn, the patients are taken from the waiting room to have a consultation with their assigned doctor, who makes a diagnosis, provides treatment and writes a prescription.

The clinic's system is coded in Java. There are many objects in this system and some of them are listed below.

Object	Description
Doctor	A licensed professional who treats patients in the clinic.
Patient	A sick person who requires a consultation with a doctor.
WaitingRoom	A place where patients wait for their consultations.
Consultation	A dated meeting between a doctor and a patient which results in a diagnosis, treatment and a prescription for medication.
Treatment	A dated record of all actions and medication prescribed to treat the patient's diagnosed condition.

The three objects Patient, WaitingRoom and Treatment have been defined in the following UML diagrams:



The Patient and WaitingRoom objects are implemented as follows:

```
public class Patient
{
    private int id;
    private String name;
    private int priority;
    private String doctor;

    public Patient(int i, String n, int p)
    {
        id = i;
        name = n;
        priority = p;
        doctor = null;
    }

    public void setId(int i) { id = i; }
    public void setName(String n) { name = n; }
    public void setPriority(int p) { priority = p; }
    public void setDoctor(String d) { doctor = d; }
    public int getId() { return id; }
    public String getName() { return name; }
    public int getPriority() { return priority; }
    public String getDoctor() { return doctor; }
    public String toString() { return id+" "+name+" "+priority+" "+doctor; }
}
```

```

public class WaitingRoom
{
    private Patient[] patients = new Patients[10];

    // uses default constructor

    public void add(Patient newPatient)
    // adds the new patient in the next empty array location
    {
        int i = 0;
        while ((patients[i] != null) && (i < 10))
        {
            i=i+1;
        }
        if (i==10) { System.out.println("No more space in the waiting room."); }
        else { patients[i] = newPatient; }
    }

    public void callNextPatient()
    // finds the next patient, outputs their details
    // and removes the patient from the array
    {
        int index = 0;
        if (patients[0]==null)
        {
            System.out.println("The waiting room is empty.");
        }
        else
        {
            index = findNextPatientIndex();
            remove(index);
        }
    }

    private int findNextPatientIndex()
    // returns the index of the first patient with the
    // highest priority in the array patients
    {
        int max = 0;
        //... code missing ...
        return max;
    }

    private void remove(int n)
    // outputs the data of the patient instance at array index n
    // and removes that patient by shifting all remaining patients
    // by one index towards the front of the array
    {
        //... code missing ...
    }
}

```

1.

- (a) Define the term *constructor*, using an example from the code on pages 14 and 15. [2]
- (b) Describe **one** additional field that might have been included in the `Patient` class. Include a data type and sample data in your answer. [2]
- (c) Describe the relationship between the `Patient` object and the `WaitingRoom` object. [2]

Consider the `WaitingRoom` class as presented on pages 14 and 15.

- (d) Construct the missing lines of code in the `findNextPatientIndex()` method to return the index of the first patient with the highest priority in the `patients` array.  
**Note:** the highest possible priority is 3. [3]
- (e) Construct the `remove(int n)` method which outputs the data of the patient object at index `n` and then removes that patient object by moving all remaining patient objects one index towards the front of the `patients` array.  
You may assume that `n` is a valid index between 0 and 9, and that an instance of `Patient` exists at that index. [6]

- (a) Award **[1]** for a definition, such as:  
A (special) method (with the same name as the class) that instantiates an object of that class;

*Award [1] for the example:*

`public Patient(String i, String n, int p)` [2]

- (b) Award **[1]** for a field and its data type.  
*For example:*

`String phoneNumber;` [2]

- (c) Award **[1]** for any indication of aggregation.  
Award an additional **[1]** for a full description.

*Example answer:*

The `WaitingRoom` object stores up to 10 instances of the `Patient` object;  
The `WaitingRoom` has a `Patient`;

Accept a correctly labelled diagram. [2]

- (d) Award marks as follows up to **[3 max]**.  
Award **[2]** for correct loop, award **[1]** for loop with one condition.  
Award **[1]** for correct test.  
Award **[1]** for assigning `max`.

*Example answer:*

```
int i=1;
while ((i<10) && (patients[i]!=null))
{
    if (patients[i].priority > patients[max].priority)
    {
        max = i;
    }
    i = i+1;
}
```

[3]



- (e) Award marks as follows up to **[6 max]**.  
Award **[1]** for including the correct signature.  
Award **[1]** for correct output of patient details.  
Award **[1]** for any loop.  
Award **[1]** for correct loop.  
Award **[1]** for correct assignment to shift a patient instance.  
Award **[1]** for assigning null.

Example answer:

```
public void remove(int n)
{
    System.out.println(patients[n].toString());
    for (int i=n; i<9; i=i+1)
    {
        patients[i] = patients[i+1];
    }
    patients[9] = null;
}
```

**[6]**

- (a) In relation to the `Patient` class, outline **one** advantage of encapsulation. [2]
- (b) In relation to the `Treatment` object, discuss **one** ethical consideration when designing software that stores patients and their illnesses. [4]

The clinic would like to start storing details in a `Doctor` object, including full name, telephone number and whether the doctor is present or not. For example:

name:	Dr Henriëtte Mănescu-Rața
phone:	0734511122
present:	true

- (c) Design the `Doctor` object using a UML diagram. [3]
- (d) In relation to the `Doctor` object, outline the need for extended character sets as used by modern programming languages. [3]

- (a) *Award [1] for a suitable definition, for example:*  
 Encapsulation means having private variables;  
 Variables not directly accessible from outside the class;  
 Methods and variables are all included in the class definition;
- Award [1] for relating an advantage to the `Patient` class, such as:*  
 So that patient data cannot be accessed/modified accidentally;  
 So that patient data is more secure; [2]

- (b) *Award [1] for identifying an ethical issue and [1] for an elaboration.*  
*Award [1] for relating this issue to the `Treatment` class and [1] for an elaboration.*

*Example answer:*

Sensitive information should be separated from identity data;  
 In order to avoid information misuse;  
 The consultation object does not include the patient's name (only an ID);  
 So that a sensitive diagnosis (like AIDS) is not directly linked to a name; [4]

- (c) *Award marks as follows up to [3 max].*  
*Award [1] for including `String name` with correct getter and setter methods.*  
*Award [1] for including `String phone` with correct getter and setter methods.*  
*Award [1] for including `Boolean present` with correct getter and setter methods;*

*Example answer:*

Doctor
String name String phone <b>Boolean</b> present
setName(String name) setPhone(String phone) setPresent(Boolean present) String getName() String getPhone() <b>Boolean</b> getPresent() String toString() //optional

[3]

- (d) Award **[1]** for identifying an appropriate need for extended character sets and an additional **[1]** for an elaboration of this need. Award **[1]** for relating to the Doctor object.

*Example answer:*

Different languages use different characters, which are not included in the basic 8-bit ASCII character set;

Extended character sets (like Unicode) include all possible characters from all languages;

For example, the doctor's name could not be spelled correctly in ASCII;

**[3]**

3.

`Treatment` objects are being instantiated throughout the day and added to a collection. The object `treatmentFile` contains the following methods which act on that collection:

- `getNext()` which reads the next treatment from the collection and returns it
- `hasNext()` which returns false when there are no more treatments in the collection.

Construct the method `showMedicationByDoctor()`, which will take the name of a doctor as a parameter and output the medication for each treatment in the collection that has been provided by that doctor. You may assume that `treatmentFile` has been declared as a global variable, that it is open for reading, and that the first time `getNext()` is called it will return the first treatment from the collection.

**[6]**

Award marks as follows up to **[6 max]**.  
Award **[1]** for correctly initialized method.  
Award **[1]** for creating a new *Treatment* object.  
Award **[1]** for loop through treatment file.  
Award **[1]** for setting *current* equal to next in file.  
Award **[2]** for correct if statement that checks names.  
Award **[1]** for correct output statement.

```
void showMedicationByDoctor(String name)
{
    Treatment current;
    while (treatmentFile.hasNext())
    {
        current = treatmentFile.getNext();
        if (current.getDoctor().equals(name))
        {
            output(current.getMedication());
        }
    }
}
```

**[6]**